Theoretical Computer Science - Bridging Course  
Winter Term 2016  
Sample Solution for Exercise Sheet 8  

Exercise 1: \(O\)-Notation Formal Proofs (2+2+2 points)  
The set \(O(f)\) contains all functions that are asymptotically not growing faster than the function \(f\) (when additive or multiplicative constants are neglected). That is:  
\[
g \in O(f) \iff \exists c \geq 0, \exists M \in \mathbb{N}, \forall n \geq M : g(n) \leq c \cdot f(n)
\]

For the following pairs of functions, check whether \(f \in O(g)\) or \(g \in O(f)\) or both. Proof your claims (you do not have to proof a negative result \(\notin\), though).

(a) \(f(n) = 100n, \ g(n) = 0.1 \cdot n^2\)
(b) \(f(n) = \log_2(n!), \ g(n) = n \log_2 n\)  
Hint: \(n! := \prod_{i=1}^{n} i \geq (n/2)^{n/2}\)
(c) \(f(n) = 2^n, \ g(n) = 3^n\)

Solution

(a) It is \(100n \in O(0.1n^2)\). To show that we require constants \(c, M\) such that \(100n \leq c \cdot 0.1n^2\) for all \(n \geq M\). Obviously this is the case for \(c = 1000\) and \(M = 1\).

(b) We have  
\[
\log_2(n!) \leq \log_2(n^n) = n \log_2 n
\]
for all \(n \geq 1\). Therefore \(\log_2(n!) \in O(n \log_2 n)\). In the other direction we have the following result  
\[
\log_2(n!) \geq \log_2((n/2)^{n/2}) = \frac{n}{2} \log_2 n - \frac{n}{2} \geq \frac{n}{2} \log_2 n - \frac{n}{4} \log_2 n = \frac{1}{4} n \log_2 n
\]
For all \(n \geq 4\). Thus \(n \log_2 n \in O(\log_2(n!))\) is also the case. In general if both \(f \in O(g)\) and \(g \in O(f)\) are true, these two functions are called asymptotically equivalent in terms of the \(O\)-notation. This is denoted by \(f \in \Theta(g)\).

(c) Obviously \(2^n \leq 3^n\) for all \(n \geq 1\). The converse is false though, because a \(c\) such that \(3^n \leq c2^n\) must fulfill \(c \geq (3/2)^n\) for arbitrarily big \(n\), but since \((3/2)^n\) is unbounded there can be no such \(c\).

Exercise 2: Sort Functions by Asymptotic Growth (4 points)  
Sort the following functions by asymptotic growth. Write \(g <_O f\) if \(g \in O(f)\). Write \(g =_O f\) if \(f \in O(g)\) and \(g \in O(f)\).

\[
\begin{array}{cccc}
n^2 & \sqrt{n} & 2^n & \log(n^2) \\
3^n & n^{100} & \log(\sqrt{n}) & (\log n)^2 \\
\log n & 10^{100} n & n! & n \log n \\
n \cdot 2^n & n^n & \sqrt{\log n} & n
\end{array}
\]
Solution

\[
<_{\log} \sqrt{n} <_{\log} \log(n) =_{\log} \log n =_{\log} \log(n^2) =_{\log} 10^{100n} <_{\log} 2^n <_{\log} n! <_{\log} n^n
\]

Exercise 3: Decision and Optimization Problems (2+2 points)

A decision problem is usually given as a formal question, whether a certain input (also called an instance) satisfies a condition or not. The language corresponding to a decision problem is the set of all instances encoded with symbols from an alphabet \(\Sigma\), for which the answer to the question is yes and that are well-formed meaning that they represent a proper instance.

Quite often a problem can also be formulated as an optimization problem, which asks for the maximum or minimum value that satisfies a certain condition. Algorithms solving decision problems can usually be used to solve the according optimization problem and vice versa. Consider the following problems:

**DominatingSet:**
- A dominating set of a graph \(G = (V, E)\) is a subset \(D \subseteq V\) such that for every vertex \(v \in V\): \(v \in D\) or \(v\) adjacent to a node \(u \in D\).
- **Input:** Encoding \(\langle G, k \rangle\) of an undirected, unweighted, simple graph \(G = (V, E)\) and \(k \in \mathbb{N}\).
- **Question:** Is there a dominating set with at most \(k\) nodes?

**VertexColoring:**
- A vertex coloring of a graph \(G = (V, E)\) is a mapping \(c : V \to \{1, \ldots, k\}\) such that \(c(u) = c(v) \Rightarrow \{u, v\} \notin E\).
- **Input:** Encoding \(\langle G \rangle\) of an undirected, unweighted, simple graph \(G = (V, E)\).
- **Question:** What is the smallest \(k\) for which a valid vertex coloring \(c\) of \(G\) exists?

(a) Are the above problems optimization or decision problems? Transform these problems into the respective other problem type.

(b) Give the languages \(k\text{-DOMINATINGSET}\) and \(k\text{-VERTEXCOLORING}\) corresponding to decision problems of the above problems.

Solution

(a) **DOMINATINGSET** is formulated as a decision problem and **VERTEXCOLORING** as an optimization problem. We rephrase the respective questions to obtain the according other problem type.

- **DOMINATINGSET:** What is the size of a smallest dominating set?
- **VERTEXCOLORING:** Is there a vertex coloring with at most \(k\) colors?

(b) \(k\text{-DOMINATINGSET} = \{\langle G \rangle \mid G \text{ has a dominating set of size at most } k.\}\)
\(k\text{-VERTEXCOLORING} = \{\langle G \rangle \mid G \text{ has a vertex coloring with at most } k \text{ colors.}\}\)

Remark: The above languages can be stated independently of \(k\) as follows

- **DOMINATINGSET** = \{\(\langle G, k \rangle \mid G \text{ has a dominating set of size at most } k.\}\)
- **VERTEXCOLORING** = \{\(\langle G, k \rangle \mid G \text{ has a vertex coloring with at most } k \text{ colors.}\}\)

Which makes these problems also much harder since the \(k\) can not be considered to be a fixed constant any more.
Exercise 4: The class $\mathcal{P}$ (1+2+2+1 points)

Clique:

- A **clique** of a graph $G = (V, E)$ is a subset $Q \subseteq V$ such that for all $u, v \in Q : \{u, v\} \in E$.

- **Input**: Encoding $\langle G, k \rangle$ of an undirected, unweighted, simple graph $G = (V, E)$ and $k \in \mathbb{N}$.

- **Question**: Is there a clique of size at least $k$?

$\mathcal{P}$ is the set of languages which can be decided by an algorithm whose runtime can be bounded by $p(n)$, where $p$ is a polynomial and $n$ the size of the respective input (problem instance). Show that the following languages ($\equiv$ problems) are in the class $\mathcal{P}$. Since it is typically easy (i.e. feasible in polynomial time) to decide whether an input is well-formed, your algorithm only needs to consider well-formed inputs. Use the $\mathcal{O}$-notation to bound the run-time of your algorithm.

(a) 1-DominatingSet
(b) 2-VertexColoring
(c) 3-Clique
(d) Any context-free language $L$.  

**Hint**: You can use results from previous exercise sheets.

**Solution**

(a) A dominating set of size one exists if and only if the input graph $G = (V, E)$ has a node with degree $|V| - 1$. Thus one possible algorithm that decides 1-DominatingSet checks for each node of $G$ if it dominates all others. For this purpose we traverse all edges in the input graph $G$ and maintain a count of the degree for all nodes. Then we check whether there is a node with maximum degree $|V| - 1$. If the result is positive we accept. Otherwise we reject $\langle G \rangle$. Since edges and nodes are traversed only a constant number of times the runtime of the algorithm is $\mathcal{O}(|E| + |V|) \subseteq \mathcal{O}(|\langle G \rangle|)$ ($|\langle G \rangle|$ is the length of the encoding of $G$). Therefore $1$-DominatingSet $\in \mathcal{P}$.

(b) The following algorithm checks whether a connected component of a graph can be colored with two colors. Since the following algorithm can simply be repeated for every component of the input graph $G = (V, E)$ we assume that $G$ is connected. We execute a Breadth First Search (BFS) on $G$ starting from an arbitrary node $s \in V$. BFS is a very basic algorithm to traverse all nodes of a graph and uses a queue. It works as follows.

Initially we color $s$ with the color 1 and enqueue all its neighbors while setting $s$ as their predecessor. Then we repeat the following steps as long the queue is not empty. Take the first node $v$ from the queue, color $v$ with the color its predecessor is not colored in. Then check whether the color of one of $v$’s colored neighbors equals $v$’s own color and if this is the case, reject $G$ and terminate. Otherwise enqueue all of $v$’s uncolored neighbors, set $v$ as their predecessor and continue. Finally, if the queue runs empty, $G$ is accepted.

A BFS explores the graph in layers, i.e. it first explores all nodes that can be reached via a single edge (hop) from the start node $s$ then all nodes with distance of exactly two hops from $s$ and so on. If $\langle G \rangle \in 2$-VertexColoring can be colored using only two colors, then the coloring is fixed by assigning an initial color to a single node $s$. This initial coloring of $s$ is subsequently completed by the BFS, which colors the layers around $s$ with alternating colors.

If $\langle G \rangle \notin 2$-VertexColoring, i.e. no 2-coloring exists for $G$, then the BFS (which explores all edges) finds an edge with equally colored end nodes and rejects $G$. Since we explore every edge at most twice and every node enters and leaves the queue exactly once, the run-time is $\mathcal{O}(|V| + |E|)$ and thus polynomial in $|\langle G \rangle|$. If $G$ consists of several components, we restart the algorithm with an arbitrary, uncolored node every time the queue runs empty, as long as uncolored nodes exist. This does not change the asymptotic run-time bound. Thus 2-VertexColoring $\in \mathcal{P}$.
(c) Let $G = (V, E)$ and $|V| = n$. Then we know $|E| = O(n^2)$. Upon input $G$, we can enumerate all possible triples $(v_1, v_2, v_3)$ such that $v_1 \neq v_2 \neq v_3 \neq v_1$. There exist at most $\binom{n}{3} = O(n^3)$ such triples. For each such triple $(v_1, v_2, v_3)$, we examine whether $(v_1, v_2) \in E$, $(v_1, v_3) \in E$, and $(v_2, v_3) \in E$. Since $|E| = O(n^2)$, this examination can be done in $O(n^2)$ time. If during the examination process, we find one triple that satisfies the requirement, we found a clique of size 3 accept $G$. Otherwise, when we finish examining all possible triple, we reject $G$ since it does not contain a clique of size 3. The runtime of the above procedure is $O(n^5)$, thus $3$-CLIQUE $\in \mathcal{P}$.

(d) Every context free language $L$ has a context-free grammar $G$ (definition) which can wlog assumed to be in Chomsky Normal Form (CNF). On exercise sheet 4 we introduced the Cocke-Younger-Kasami (CYK) algorithm which solves the word problem ($s \in L(G)$?) for grammar $G$ in CNF and any input string $s \in \Sigma^*$ in $O(|s|^3)$. Thus $L \in \mathcal{P}$. 

4